

Criptografia i Seguretat
Laboratori
Curs 2024-25
UAB

Laboratori 1:
**Implementació de AES
en Python**

(Data: 2025-03-25)

David Morillo Massagué
Adrià Muro Gómez

Taula de continguts

Introducció.....	3
Objectius.....	3
Fonaments Teòrics.....	3
Algorisme AES.....	3
Implementació.....	4
Estructura general del xifrat.....	4
Funcions Auxiliars.....	6
Funcions de Transformació.....	10
AddRoundKey.....	10
SubBytes.....	11
ShiftRows.....	13
MixColumns.....	14
Resultats i proves.....	17
Resultat de l'execució.....	17
Validació de l'algoritme.....	17
6. Conclusions.....	20
7. Bibliografia.....	21
8. Annexos.....	21

Introducció

En aquesta pràctica s'ha implementat l'algorisme de xifratge AES (Advanced Encryption Standard) amb l'objectiu de comprendre el seu funcionament intern i validar-ne el correcte comportament a través de proves. Es destaquen les operacions bàsiques de l'algorisme, com SubBytes, ShiftRows, MixColumns i AddRoundKey, per garantir la seguretat de la informació.

Objectius

L'objectiu principal d'aquesta pràctica és dissenyar i implementar una versió funcional de l'algorisme AES en Python, assegurant que el seu comportament sigui correcte mitjançant proves exhaustives. Per assolir aquest objectiu general, es defineixen els següents objectius específics:

- **Comprendre el funcionament intern de l'AES:** estudiar els passos que conformen el procés de xifratge i desxifratge, incloent-hi les operacions de SubBytes, ShiftRows, MixColumns i AddRoundKey.
 - **Implementar l'algorisme de forma eficient:** escriure un codi optimitzat que compleixi els estàndards de seguretat i rendiment, evitant vulnerabilitats comunes.
 - **Validar el correcte funcionament de la implementació:** realitzar proves amb diferents claus i textos en clar per assegurar que el xifratge i el desxifratge es duen a terme correctament.
-

Fonaments Teòrics

Algorisme AES

AES (Advanced Encryption Standard) és un algorisme de xifrat simètric àmpliament utilitzat per protegir la confidencialitat de les dades. Com a algorisme simètric, AES utilitza la mateixa clau per xifrar i desxifrar les dades. AES es va establir com a estàndard pel NIST (Institut Nacional d'Estàndards i Tecnologia) el 2001, substituint l'antic algorisme DES (Data Encryption Standard).

AES pot utilitzar claus de 128 bits, 192 bits o 256 bits, amb més rondes de xifrat per a claus més llargues (10, 12 o 14 rondes, respectivament). Treballa amb blocs de 128 bits (16 bytes), i cada ronda aplica transformacions a aquests blocs.

Les transformacions als blocs es realitzen a partir d'un conjunt d'operacions:

- **SubBytes:** Substitució de cada byte del bloc amb un altre valor mitjançant una taula de substitució (S-Box).
- **ShiftRows:** Desplaçament cíclic de les files del bloc de dades.
- **MixColumns:** Barreja de les columnes del bloc de dades per millorar la difusió.
- **AddRoundKey:** XOR entre el bloc de dades i una clau derivada de la clau original.

AES és conegut per la seva seguretat i eficiència, amb una alta resistència als atacs de força bruta i altres tècniques d'atac criptogràfic. El seu ús és àmpliament recomanat en aplicacions que requereixen protecció de dades sensibles, com la comunicació en línia, sistemes de pagament i emmagatzematge segur.

Implementació

Estructura general del xifrat

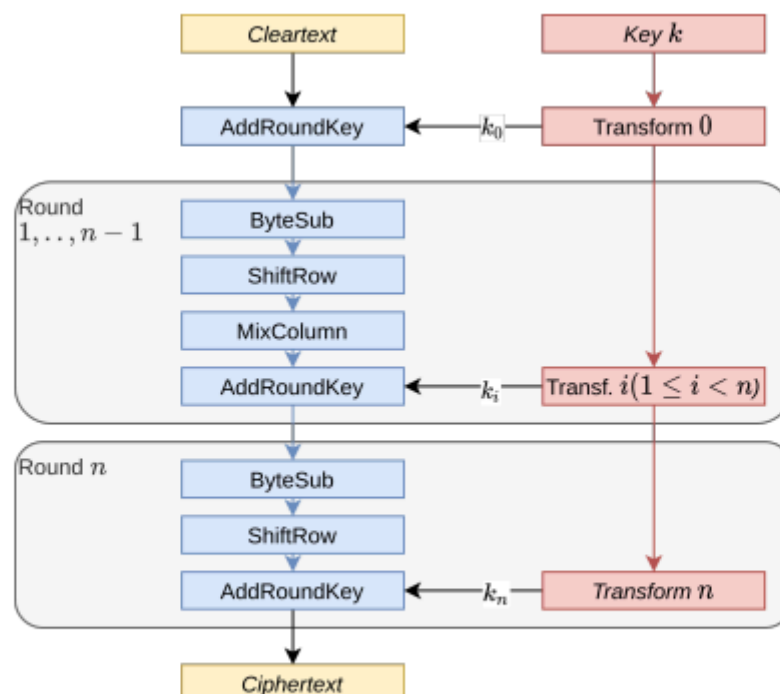


Figure 1: Operacions de xifrat de l'AES.

La figura anterior mostra la representació gràfica de l'estructura general del procés de xifrat AES. El missatge en clar (clear text) passa primer per l'operació inicial d'Add Round Key on es combina amb la clau de la primera ronda.

Després, es realitzen múltiples rondes que inclouen les operacions de SubBytes, ShiftRows, MixColumns i Add Round Key, iterant un total de 9 vegades. Finalment, es realitza una darrera ronda sense l'operació de MixColumns, on es torna a aplicar SubBytes, ShiftRows i l'últim Add Round Key per obtenir el text xifrat.

La nostra manera de treballar ha consistit en descompondre les operacions en funcions individuals per a cada acció necessària, com són la substitució de bytes (SubBytes), la permutació de files (ShiftRows), la multiplicació de Galois (MixColumns) i l'addició de la clau (AddRoundKey).

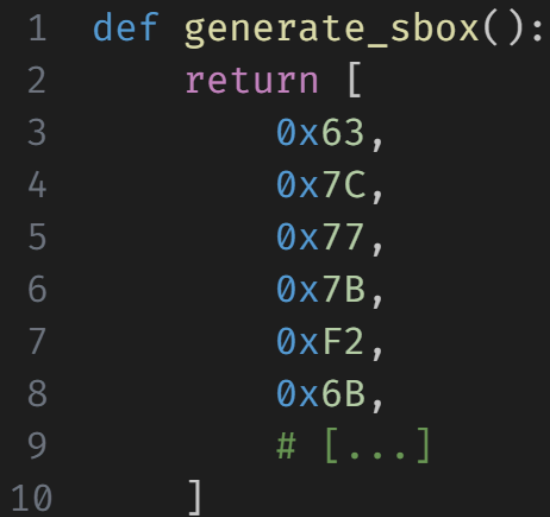
Un cop les funcions estan definides, s'han agrupat en una funció principal (aes_encrypt) que implementa les 10 rondes de xifrat tal com descriu l'algorisme AES. Així, cada component és modular i fàcil de gestionar.

S'ha decidit separar les funcions en dos grups:

- Funcions Auxiliars
 - Funcions de Transformació
-

Funcions Auxiliars

- **Generació de la S-Box (generate_sbox)**

A screenshot of a code editor with a dark background and light-colored text. The code is a Python function definition for generate_sbox. It starts with a line number 1, followed by 'def generate_sbox():'. Line 2 has 'return [' followed by a list of hexadecimal values: '0x63,' on line 3, '0x7C,' on line 4, '0x77,' on line 5, '0x7B,' on line 6, '0xF2,' on line 7, and '0x6B,' on line 8. Line 9 has a comment '# [...]' and line 10 closes the list with a closing bracket ']'.

```
1 def generate_sbox():
2     return [
3         0x63,
4         0x7C,
5         0x77,
6         0x7B,
7         0xF2,
8         0x6B,
9         # [...]
10    ]
```

Aquesta funció genera la S-Box (Substitution Box), que és un component fonamental d'AES. La S-Box es fa servir per a la substitució de bytes en cada ronda de xifrat. Es basa en una taula de 256 valors, on cada valor es correspon amb un byte de la clau. Aquesta substitució proporciona una mica de "complexitat" o "no-linealitat" al procés de xifrat, augmentant la seguretat.

- **Generació de Rcon (generate_rcon)**

```
1 def generate_rcon():
2     return [
3         0x00000000,
4         0x01000000,
5         0x02000000,
6         0x04000000,
7         0x08000000,
8         0x10000000,
9         0x20000000,
10        0x40000000,
11        0x80000000,
12        0x1B000000,
13        0x36000000,
14    ]
```

La funció `generate_rcon` genera els valors de Rcon (Round Constant), que són valors utilitzats durant l'expansió de la clau per afegir més variabilitat a les claus de cada ronda. Els valors de Rcon s'utilitzen per a modificar la clau de cada ronda i garantir que les claus de cada ronda siguin úniques i segures.

- **Multiplicació per Galois (galois_mult)**

```
1 def galois_mult(x, y):
2     if x == 1:
3         return y % 256
4     elif x == 2:
5         resultado = y << 1
6         if y & 0x80:
7             resultado ^= 0x1B
8         return resultado % 256
9     elif x == 3:
10        return galois_mult(1, y) ^ galois_mult(2, y)
11    else:
12        return None
```

La funció `galois_mult` implementa la multiplicació en el camp de Galois, una operació matemàtica que es fa servir a l'algorisme AES, especialment en la transformació de les columnes de l'estat (mixing columns). AES utilitza aquesta operació per modificar els valors de cada columna en el bloc de text xifrat, aportant més complexitat al procés de xifrat.

- **Expansió de la clau (`key_expansion`)**

```
1 def key_expansion(key, sbox, Rcon, round_num):
2     def rotword(word):
3         return word[1:] + word[:1]
4
5     def subword(word):
6         return [sbox[b] for b in word]
7
8     def rcon(i):
9         if i < len(Rcon):
10            return [Rcon[i] >> 24, 0x00, 0x00, 0x00]
11            return [0x00, 0x00, 0x00, 0x00]
12
13     expanded_keys = np.array([[int(x, 16) for x in col] for col in zip(*key)]).tolist()
14
15     new_key = []
16     new_column = []
17
18     temp = expanded_keys[-1]
19
20     temp = subword(rotword(temp))
21     temp = [temp[j] ^ rcon(round_num)[j] for j in range(4)]
22     temp = [temp[j] ^ expanded_keys[0][j] for j in range(4)]
23
24     new_key.append(temp)
25
26     for i in range(1, 4):
27
28         new_column = [new_key[-1][j] ^ expanded_keys[i][j] for j in range(4)]
29         new_key.append(new_column)
30
31     new_key_hex = [[f"0x{num:02x}" for num in col] for col in new_key]
32     new_key_hex = [list(x) for x in zip(*new_key_hex)]
33
34     return new_key_hex
```

La funció `key_expansion` genera una sèrie de claus de ronda a partir de la clau inicial proporcionada. AES utilitza un procés d'expansió per generar les claus necessàries per a cada ronda de xifrat. Per cada ronda, es realitzen operacions de substitució, rotació i aplicació de constants `Rcon` a la clau anterior per generar la clau de la següent ronda.

- Cifrat AES (aes_encrypt)

```
1 def aes_encrypt(estado, key):
2
3     sbox = generate_sbox()
4     Rcon = generate_rcon()
5
6     # Ronda inicial
7     estado = add_round_key(estado, key)
8
9     # 9 rondas principales
10    for i in range(1, 10):
11        # SubBytes
12        estado = subbytes(sbox, estado)
13
14        # ShiftRows
15        estado = shiftrows(estado)
16
17        # MixColumns
18        estado = mixcolumns(estado)
19
20        # Expandir clave
21        key = key_expansion(key, sbox, Rcon, i)
22
23        # AddRoundKey
24        estado = add_round_key(estado, key)
25
26    # Ronda final
27    estado = subbytes(sbox, estado)
28    estado = shiftrows(estado)
29    key = key_expansion(key, sbox, Rcon, 10)
30    estado = add_round_key(estado, key)
31
32    return estado
```

Aquesta funció implementa el procés global d'enciptació AES. Inicialment, es fa un "Add Round Key" per afegir la clau de la primera ronda. A continuació, s'apliquen 9 rondes de substitució de bytes, desplaçament de files, mescla de columnes i afegir la clau de ronda. Finalment, es realitza una ronda final, on només s'aplica la substitució de bytes, el desplaçament de files i l'afegir la clau de ronda (sense la mescla de columnes). El resultat final és el text xifrat.

Funcions de Transformació

AddRoundKey

Matriu d'estat d'entrada:

```
Inicial AddRoundKey - Input:
+-----+-----+-----+-----+
| 0x4d | 0x61 | 0x53 | 0x65 |
+-----+-----+-----+-----+
| 0x69 | 0x74 | 0x65 | 0x74 |
+-----+-----+-----+-----+
| 0x73 | 0x67 | 0x63 | 0x20 |
+-----+-----+-----+-----+
| 0x73 | 0x65 | 0x72 | 0x20 |
+-----+-----+-----+-----+
```

Matriu d'estat de sortida:

```
Inicial AddRoundKey - Output:
+-----+-----+-----+-----+
| 0xe  | 0x32 | 0x36 | 0x45 |
+-----+-----+-----+-----+
| 0x5  | 0x11 | 0x11 | 0x54 |
+-----+-----+-----+-----+
| 0x12 | 0x4  | 0x2  | 0x0  |
+-----+-----+-----+-----+
| 0x6  | 0x17 | 0x52 | 0x0  |
+-----+-----+-----+-----+
```

Descripció de l'operació:

L'operació AddRoundKey consisteix a aplicar una operació XOR (exclusiu OR) entre la matriu d'estat i la clau d'aquesta ronda. A l'algorisme AES, la clau s'expandeix per obtenir una clau per cada ronda, i aquesta clau es combina amb l'estat actual de les dades per a cada ronda.

Càlcul realitzat:

```
1 def add_round_key(estado, clave):
2     for i in range(4):
3         for j in range(4):
4             estado[i][j] = hex(int(estado[i][j], 16) ^ int(clave[i][j], 16))
5     return estado
```

A la funció `add_round_key`, es realitza una operació de XOR entre la matriu d'estat (estado) i la matriu de la clau (clave) en cada cel·la. Aquesta operació és una de les principals operacions de la transformació de xifratge AES, i consisteix en el següent:

- Operació de XOR:
 - Per cada element de la matriu estado, es realitza un XOR entre el valor del byte d'aquesta matriu i el valor corresponent del byte de la matriu de la clau clave.
 - El resultat d'aquesta operació es converteix a una cadena hexadecimal.
- Detalls del càlcul:
 - Cada valor de les matrius estado i clave es converteix primer en un enter a partir de la seva representació hexadecimal mitjançant `int(estado[i][j], 16)` i `int(clave[i][j], 16)`.
 - S'aplica l'operació XOR sobre aquests dos valors enters, i el resultat es torna a convertir en hexadecimal mitjançant `hex(...)`.
 - Aquesta operació s'aplica a tots els elements de la matriu d'estat.
- Implementació:
 - La funció utilitza dues dimensions (4x4) per recórrer les matrius estado i clave, i en cada iteració, l'operació XOR es realitza sobre els elements corresponents de les dues matrius. Així, es modifica la matriu estado amb els resultats de les operacions XOR.

SubBytes

Matriu d'estat d'entrada:

SubBytes - Input:				
0xe	0x32	0x36	0x45	
0x5	0x11	0x11	0x54	
0x12	0x4	0x2	0x0	
0x6	0x17	0x52	0x0	

Matriu d'estat de sortida:

SubBytes - Output:			
0xab	0x23	0x5	0x6e
0x6b	0x82	0x82	0x20
0xc9	0xf2	0x77	0x63
0x6f	0xf0	0x0	0x63

Descripció de l'operació:

SubBytes és una operació de substitució no lineal on cada byte de la matriu d'estat es reemplaça amb un valor corresponent en una taula fixa de substitució anomenada **S-Box**. Aquesta taula es genera a partir d'un procés específic que garanteix que l'algorisme sigui resistent a atacs de tipus "diferencial" i "lineal".

Càlcul realitzat:

```
1 def subbytes(s_box, estado):
2     for i in range(4):
3         for j in range(4):
4             estado[i][j] = hex(s_box[int(estado[i][j], 16)])
5     return estado
```

A la funció subbytes, s'aplica una transformació a cada byte de la matriu d'estat mitjançant la substitució utilitzant una taula de substitució anomenada S-box. Aquest procés és una part essencial de l'algorisme AES, i es realitza de la següent manera:

- Operació de substitució:
 - La funció recorre cada byte de la matriu d'estat (una matriu de 4x4).
 - Cada element de la matriu (que es troba en forma hexadecimal) es substitueix mitjançant la taula S-box.
 - Es realitza la substitució utilitzant l'índex del valor hexadecimal del byte actual a la taula S-box.
- Detalls del càlcul:

- Es converteix el valor de cada byte de la matriu estado a un nombre enter a partir de la seva representació hexadecimal utilitzant `int(estado[i][j], 16)`.
 - Aquest nombre s'utilitza com a índex per cercar el valor corresponent a la S-box (`s_box[...]`).
 - El valor substituït es converteix de nou en una cadena hexadecimal amb `hex(...)`.
 - Implementació:
 - Aquest càlcul es fa de forma iterativa, recorrent la matriu estado i aplicant la substitució a cada element. La funció `s_box[int(estado[i][j], 16)]` retorna el valor substituït per a cada byte de la matriu.
-

ShiftRows

Matriu d'estat d'entrada:

```
ShiftRows - Input:
+-----+-----+-----+-----+
| 0xab | 0x23 | 0x5  | 0x6e |
+-----+-----+-----+-----+
| 0x6b | 0x82 | 0x82 | 0x20 |
+-----+-----+-----+-----+
| 0xc9 | 0xf2 | 0x77 | 0x63 |
+-----+-----+-----+-----+
| 0x6f | 0xf0 | 0x0  | 0x63 |
+-----+-----+-----+-----+
```

Matriu d'estat de sortida:

```
ShiftRows - Output:
+-----+-----+-----+-----+
| 0xab | 0x23 | 0x5  | 0x6e |
+-----+-----+-----+-----+
| 0x82 | 0x82 | 0x20 | 0x6b |
+-----+-----+-----+-----+
| 0x77 | 0x63 | 0xc9 | 0xf2 |
+-----+-----+-----+-----+
| 0x63 | 0x6f | 0xf0 | 0x0  |
+-----+-----+-----+-----+
```

Descripció de l'operació:

ShiftRows és una operació en què les files de la matriu d'estat es desplacen de manera cíclica. La primera fila no es mou, però les altres files es desplacen una posició a l'esquerra en cada ronda successiva. Aquesta operació ajuda a difondre les dades a través de les files.

Càlcul realitzat:

```
1 def shiftrows(estado):
2     for i in range(4):
3         estado[i] = estado[i][i:] + estado[i][:i] # Mueve i posiciones a la izquierda
4     return estado
```

A la funció shiftrows, l'objectiu és aplicar la permutació de les files de la matriu d'estat seguint les regles de l'algorisme AES. Aquesta operació consisteix en desplaçar les files de la matriu d'estat a l'esquerra segons el seu índex de fila.

- Operació de desplaçament:
 - La funció recorre les quatre files de la matriu (representades per l'índex i).
 - Per a cada fila, s'aplica un desplaçament circular a l'esquerra de i posicions, és a dir, la fila es divideix en dues parts: la part de davant (primeres i posicions) es mou al final de la fila.
 - Implementació:
 - Aquest desplaçament es realitza mitjançant la tècnica de l'operació de llistes en Python, utilitzant la sintaxi `estado[i][i:] + estado[i][:i]`, que agafa la part de la fila des de la posició i fins al final i la combina amb la part des del principi fins a la posició i.
-

MixColumns

Matriu d'estat d'entrada:

```
MixColumns - Input:
+-----+-----+-----+-----+
| 0xab | 0x23 | 0x5  | 0x6e |
+-----+-----+-----+-----+
| 0x82 | 0x82 | 0x20 | 0x6b |
+-----+-----+-----+-----+
| 0x77 | 0x63 | 0xc9 | 0xf2 |
+-----+-----+-----+-----+
| 0x63 | 0x6f | 0xf0 | 0x0  |
+-----+-----+-----+-----+
```

Matriu d'estat de sortida:

```
MixColumns - Output:
+-----+-----+-----+-----+
| 0xc4 | 0xd7 | 0x53 | 0x93 |
+-----+-----+-----+-----+
| 0x4e | 0xf6 | 0xf5 | 0xb5 |
+-----+-----+-----+-----+
| 0x62 | 0xd6 | 0xa7 | 0xfa |
+-----+-----+-----+-----+
| 0xd5 | 0x5a | 0x1d | 0x2b |
+-----+-----+-----+-----+
```

Descripció de l'operació:

MixColumns és una operació que transforma les columnes de la matriu d'estat mitjançant una combinació lineal d'elements en cada columna. L'objectiu és difondre més les dades, garantint que cada bit de la sortida depengui de tots els bits de l'entrada. Aquesta operació fa servir una matriu fixa per fer la multiplicació.

Càlcul realitzat:

```
1 def mixcolumns(estado):
2
3     for i in range(4):
4         a = estado[0][i]
5         b = estado[1][i]
6         c = estado[2][i]
7         d = estado[3][i]
8
9         a, b, c, d = int(a, 16), int(b, 16), int(c, 16), int(d, 16)
10
11         estado[0][i] = hex((galois_mult(2, a) ^ galois_mult(3, b) ^ c ^ d) % 256)
12         estado[1][i] = hex((a ^ galois_mult(2, b) ^ galois_mult(3, c) ^ d) % 256)
13         estado[2][i] = hex((a ^ b ^ galois_mult(2, c) ^ galois_mult(3, d)) % 256)
14         estado[3][i] = hex((galois_mult(3, a) ^ b ^ c ^ galois_mult(2, d)) % 256)
15
16     return estado
```

A la funció mixcolumns, es realitza una operació matemàtica sobre les columnes de la matriu d'estat (estado) mitjançant l'ús de la multiplicació en el camp de Galois. Aquesta operació transforma cada columna de la matriu de 4 elements segons una fórmula específica. El procés es realitza de la següent manera:

- Operació de multiplicació en Galois:
 - Cada element de la columna es multiplica per constants específiques del xifratge AES utilitzant la multiplicació en el camp de Galois.
 - Les constants utilitzades per la multiplicació són 2 i 3, com es defineix en el mètode galois_mult.
- Detalls del càlcul:
 - Cada columna de la matriu d'estat està formada per 4 elements: a, b, c, i d.
 - S'aplica la fórmula següent a cada element de la columna:
 - Per estado[0][i]: $(\text{galois_mult}(2, a) \wedge \text{galois_mult}(3, b) \wedge c \wedge d) \% 256$
 - Per estado[1][i]: $(a \wedge \text{galois_mult}(2, b) \wedge \text{galois_mult}(3, c) \wedge d) \% 256$
 - Per estado[2][i]: $(a \wedge b \wedge \text{galois_mult}(2, c) \wedge \text{galois_mult}(3, d)) \% 256$
 - Per estado[3][i]: $(\text{galois_mult}(3, a) \wedge b \wedge c \wedge \text{galois_mult}(2, d)) \% 256$
 - Aquesta fórmula implica:

- Multiplicacions per 2 i 3 (operacions sobre el camp de Galois).
 - Operacions XOR entre els resultats de les multiplicacions i els altres elements de la columna.
 - El resultat final de cada operació es redueix a un valor entre 0 i 255 mitjançant l'operació % 256.
- Implementació:
 - Els valors de la matriu estado es tracten com a cadenes hexadecimals, que es converteixen primer a enters per realitzar les operacions matemàtiques.
 - Després de realitzar els càlculs per a cada element de la columna, el resultat es converteix novament a hexadecimal i s'assigna a la matriu d'estat.
-

Resultats i proves

Resultat de l'execució

Després d'aplicar l'algorisme AES per al xifratge del missatge, hem obtingut el ciphertext resultant. Aquest ciphertext representa la versió encriptada de la informació original, la qual no és directament interpretable sense la clau de desxifrat corresponent. L'input utilitzat en el nostre exemple ha estat:

- Missatge: MissatgeSecret
- Clau: ClauSecreta

Missatge sense xifrar (hexadecimal):

```
Missatge Xifrat - Input:
+-----+-----+-----+-----+
| 0x4d | 0x61 | 0x53 | 0x65 |
+-----+-----+-----+-----+
| 0x69 | 0x74 | 0x65 | 0x74 |
+-----+-----+-----+-----+
| 0x73 | 0x67 | 0x63 | 0x20 |
+-----+-----+-----+-----+
| 0x73 | 0x65 | 0x72 | 0x20 |
+-----+-----+-----+-----+
```

Missatge xifrat obtingut (hexadecimal):

```
Missatge Xifrat - Output:
+-----+-----+-----+-----+
| 0x1f | 0x12 | 0xa1 | 0x3  |
+-----+-----+-----+-----+
| 0xbb | 0x42 | 0xfa | 0xe4 |
+-----+-----+-----+-----+
| 0xab | 0x3c | 0xe3 | 0x85 |
+-----+-----+-----+-----+
| 0xee | 0xe4 | 0xe3 | 0x3f |
+-----+-----+-----+-----+
```

Validació de l'algoritme

En aquesta secció es valida que la implementació del nostre xifratge AES funciona correctament. Per fer-ho, hem utilitzat la llibreria PyCryptodome, una eina reconeguda per a criptografia en Python, per comparar els resultats obtinguts amb la nostra pròpia implementació.

Per assegurar-nos que el nostre sistema de xifrat funciona correctament, hem seguit els següents passos:

1. Generació del missatge i la clau: Hem utilitzat un missatge de prova i una clau de 16 bytes, adequats per AES-128.
2. Conversió a matriu hexadecimal: La nostra implementació transforma el text en una matriu de 4x4 valors hexadecimal, seguint l'estructura de l'AES.
3. Xifratge amb la nostra implementació: Hem aplicat el nostre algorisme de xifrat per obtenir el text xifrat.
4. Xifratge amb PyCryptodome: Hem realitzat el mateix procés amb PyCryptodome per comparar-ne els resultats.
5. Comparació de resultats: Si el text xifrat generat per la nostra implementació coincideix amb el de PyCryptodome, confirmem que el nostre algorisme és correcte.
6. Desxifratge i verificació: Desxifrem el text xifrat i comprovem que es recupera exactament el missatge original.

PyCryptodome és una llibreria criptogràfica avançada que proporciona una implementació segura i eficient de AES. Hem utilitzat el mode AES-ECB per realitzar la comparació amb el nostre xifrat.

Els passos que hem seguit amb PyCryptodome són:

1. Creació d'un objecte AES amb la clau.

```
cipher = AES.new(key_bytes, AES.MODE_ECB)
```
2. Conversió de la matriu de dades a bytes.

```
estado_bytes = mtxhex2bytes(estado)
```
3. Aplicació de la funció encrypt() per obtenir el missatge xifrat.

```
estado_cifrado_bytes = cipher.encrypt(estado_bytes)  
estado_cifrado = bytes2mtxhex(estado_cifrado_bytes)
```
4. Aplicació de decrypt() sobre el missatge xifrat per recuperar el text original.

```
cipher_dec = AES.new(key_bytes, AES.MODE_ECB)  
estado_desxifrat_bytes = cipher_dec.decrypt(estado_cifrado_bytes)  
estado_desxifrat = bytes2mtxhex(estado_desxifrat_bytes)
```
5. Aplicació de mtxhex2str() per convertir el missatge desxifrat d'hexadecimal al original.

```
mensaje_desxifrat = mtxhex2str(estado_desxifrat)  
print("\nText desxifrat:", mensaje_desxifrat)
```

Comparacions entre xifrat pròpi i xifrat amb PyCryptodome:

Xifrat Pròpi	Xifrat amb PyCryptodome
<pre>Missatge Xifrat - Input: +-----+-----+-----+-----+ 0x4d 0x61 0x53 0x65 +-----+-----+-----+-----+ 0x69 0x74 0x65 0x74 +-----+-----+-----+-----+ 0x73 0x67 0x63 0x20 +-----+-----+-----+-----+ 0x73 0x65 0x72 0x20 +-----+-----+-----+-----+</pre>	<pre>Missatge - Input: +-----+-----+-----+-----+ 0x4d 0x61 0x53 0x65 +-----+-----+-----+-----+ 0x69 0x74 0x65 0x74 +-----+-----+-----+-----+ 0x73 0x67 0x63 0x20 +-----+-----+-----+-----+ 0x73 0x65 0x72 0x20 +-----+-----+-----+-----+</pre>

<pre>Missatge Xifrat - Output: +-----+-----+-----+-----+ 0x1f 0x12 0xa1 0x3 +-----+-----+-----+-----+ 0xbb 0x42 0xfa 0xe4 +-----+-----+-----+-----+ 0xab 0x3c 0xe3 0x85 +-----+-----+-----+-----+ 0xee 0xe4 0xe3 0x3f +-----+-----+-----+-----+</pre>	<pre>Missatge Xifrat PyCryptodome - +-----+-----+-----+-----+ 0x1f 0x12 0xa1 0x3 +-----+-----+-----+-----+ 0xbb 0x42 0xfa 0xe4 +-----+-----+-----+-----+ 0xab 0x3c 0xe3 0x85 +-----+-----+-----+-----+ 0xee 0xe4 0xe3 0x3f +-----+-----+-----+-----+</pre>
--	---

Matriu hexadecimal i la seva conversió a text per verificar que la desxifració funciona:

```
Missatge Desxifrat Pycryptodome - Output:
+-----+-----+-----+-----+
| 0x4d | 0x61 | 0x53 | 0x65 |
+-----+-----+-----+-----+
| 0x69 | 0x74 | 0x65 | 0x74 |
+-----+-----+-----+-----+
| 0x73 | 0x67 | 0x63 | 0x20 |
+-----+-----+-----+-----+
| 0x73 | 0x65 | 0x72 | 0x20 |
+-----+-----+-----+-----+

Text desxifrat: MissatgeSecret
```

6. Conclusions

L'anàlisi dels resultats obtinguts ens permet concloure que la implementació de l'algorisme AES desenvolupada en aquest treball és correcta i funcional. Les proves realitzades han permès verificar diversos aspectes fonamentals:

- **El nostre algorisme retorna exactament el mateix resultat que PyCryptodome en el xifrat.** Això demostra que la transformació de dades mitjançant les diferents rondes d'AES es realitza correctament, seguint l'estàndard.
- **El procés de desxifrat retorna correctament el text original.** Això confirma que l'aplicació inversa de les operacions d'AES (SubBytes, ShiftRows, MixColumns i AddRoundKey) funciona de manera adequada.
- **Hem verificat que la matriu hexadecimal generada pel nostre codi correspon a la representació en bytes utilitzada per PyCryptodome.** Això assegura que la nostra conversió de text a matriu i la seva posterior interpretació són correctes.

Aquestes validacions són essencials per garantir que l'algorisme implementat pot ser utilitzat en aplicacions reals de seguretat sense errors en el xifrat o desxifrat. En resum, la nostra implementació segueix fidelment el funcionament esperat d'AES i els resultats obtinguts avalen la seva fiabilitat.

En aquesta pràctica hem aprofundit en un dels algorismes més coneguts i robustos de la criptografia moderna: l'AES. Això ens ha permès explorar i experimentar amb els seus processos de substitució i permutació, observant com cada pas transforma el missatge d'una manera que aparentment sembla irreversible. No obstant això, hem pogut comprovar que, amb l'ús de llibreries externes, aquests passos es poden desfer amb precisió. Inicialment, ens va sorprendre la quantitat de transformacions aplicades a només uns quants bytes i la complexitat aparent del procés de desxifrat, fins i tot coneixent la clau utilitzada. Aquesta pràctica combina molts exercicis criptogràfics alhora, resultant en una activitat molt completa.

7. Bibliografia

- Rijndael_Animation. (s. f.).
https://formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng-html5.html
- Pyca. (s. f.). *GitHub - pyca/cryptography: cryptography is a package designed to expose cryptographic primitives and recipes to Python developers.* GitHub.
<https://github.com/pyca/cryptography>
- Legrandin. (s. f.). *GitHub - Legrandin/pycryptodome: A self-contained cryptographic library for Python.* GitHub. <https://github.com/Legrandin/pycryptodome>
- CrypTool Portal. (s. f.). CrypTool Portal.
<https://www.cryptool.org/en/cto/aes-step-by-step>
- *m2crypto / m2crypto · GitLab.* (s. f.). GitLab. <https://gitlab.com/m2crypto/m2crypto>
- colaboradores de Wikipedia. (2025, 26 enero). *Advanced Encryption Standard.* Wikipedia, la Enciclopedia Libre.
https://es.wikipedia.org/wiki/Advanced_Encryption_Standard
- Cilleruelo, C. (2024, 18 abril). ¿Qué es el algoritmo AES? | KeepCoding Bootcamps. *KeepCoding Bootcamps.* <https://keepcoding.io/blog/que-es-el-algoritmo-aes/>

8. Annexos

- Enllaç al codi utilitzat a la pràctica:
https://github.com/Dakuur/criptografia/blob/main/lab1/lab1_cripto.ipynb